Examining the last writer slices from failing and non-failing executions reveals the ordering violation.

Thread 1

```
void hts_mutexinit(htsmutex* mutex) {
   htsmutex_s* smutex = malloct(sizeof(htsmutex_s));
   pthread_mutex_init(&smutex, 0);
A  *mutex = smutex
}
```

Bug: Thread 2 uses the htsmutex before Thread 1 initializes it.

Thread 2

```
int hts_cancel_file_push(...) {
   int ret;
B  hts_mutexlock(&opt->state.lock);
   ret = hts_cancel_file_push_(opt, url);
   hts_mutexrelease(&opt->state.lock);
   return ret; }
```

Non-Failing Run Memory Image @ B

&opt->state.lock = <valid mutex state>

Failing Run Core dump @ B

&opt->state.lock = <garbage data> or NULL

Non-Failing Run Last Writer Slice @ B

LWS(&opt->state.lock) = Thread 1@ A

Failing Run Last Writer Slice @ B

LWS(&opt->state.lock) = <uninitialized>